IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON DATA MINING AND GRANULAR COMPUTING IN BIG DATA AND KNOWLEDGE PROCESSING

Received October 6, 2018, accepted October 28, 2018, date of publication November 9, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878897

A Multimodel Fusion Engine for Filtering Webpages

ZIYUN DENG[1,2], TINGQIN HE[2], WEIPING DING[3,4], (Member, IEEE), AND ZEHONG CAO[4], (Member, IEEE)

[1]College of Economics and Trade, Changsha Commerce and Tourism College, Changsha 410116, China
[2]National Supercomputing Center in Changsha, Hunan University, Changsha 410082, China
[3]School of Computer Science and Technology, Nantong University, Nantong 226019, China
[4]Centre for Artificial Intelligence, Faculty of Engineering and Information Technologies, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Ziyun Deng (dengziyun@126.com)

This work was supported in part by the National Key Technology Support Program of China under Grant 2012BAH09B02, in part by the Natural Science Foundation of Hunan Province under Grant 2017JJ5064, in part by the Social Science Foundation of Hunan Province under Grant 16ZDA07, and in part by the Applied Basic Research Program of Nantong under Grant GY12016014.

ABSTRACT Fusing multiple existing models for filtering webpages can mitigate the shortcomings of individual filtering models. To provide an engine for such fusion, we propose a multimodel fusion engine for filtering webpages for the extraction of target webpages. This engine can handle large datasets of webpages crawled from websites and supports five individual filtering models and the fusion of any two of them. There are two possible fusion methods: one is to simultaneously satisfy the conditions of both individual models, and the other is to satisfy the conditions of one of the two individual models. We present the functions, architecture, and software design of the proposed engine. We use recall ratio (RR) and precision ratio (PR) as the evaluation indices of the filtering models and propose rules describing how PR and RR change when individual models are fused. We use 200 000 webpages collected by crawling the popular online shopping website "www.jd.com" as the experimental dataset to verify these rules. The experimental results show that two-model fusion can improve either PR or RR. Thus, the proposed engine has good practical value for engineering applications.

INDEX TERMS Multimodel, fusion, engine design, webpage filtering.

I. INTRODUCTION

An engine for filtering webpages is an important component of a search engine. Its purpose is to screen incoming webpages to determine which pages should be displayed to users. A good webpage filtering engine allows users to block pages from websites that are likely to include unwanted advertising, pornographic content, spyware, viruses, or other objectionable content. The academic and engineering communities have developed various webpage filtering engines that support one or more filtering models.

At present, most vertical web crawlers support only a single filtering model. Some researchers have fused 2–3 individual filtering models to obtain new filtering models [1]–[4], thereby proving that fusing existing filtering models is an effective way to mitigate the shortcomings of individual filtering models [1]–[3]. However, no engine that supports multiple individual filtering models and their fusion is available on the market. Therefore, to fill this gap, we developed a multimodel fusion engine for filtering webpages (MMFEFWP). MMFEFWP can handle large datasets from webpages crawled from websites and supports five individual filtering models and the fusion of any two of them.

II. RELATED WORK

Four categories of webpage filtering models have been proposed in existing research [4], [5]. The first category consists of filtering models based on the uniform resource identifiers (URIs) of webpages [6]. The second consists of filtering models based on the features of webpage tags. The third consists of filtering models based on the structures of webpages. The fourth consists of filtering models based on autonomous learning. These existing models also aim to delete redundant tags from the webpages before filtering.

Filtering models based on the URIs of webpages are simple and easy to implement, but they are not suitable for the new mechanism of URI generation and mapping used by many

websites. For convenience, many websites map their URIs to shorter URIs. Using these shorter URIs alone, models of this kind cannot determine whether a given webpage is a target page [6].

Filtering models based on the features of webpage tags use several essential features to define the filtering conditions of webpages, including specific strings, specific tags [7], [8], specific tree nodes [9], specific tree structures in a webpage [10], and the link ratio of a webpage. These filtering models are applicable only to specific websites and require programmers to have a prior understanding of the content of these websites [11], [12].

Filtering models based on the structures of webpages can be divided into two subclasses. The models in one subclass filter webpages by calculating the similarities between two XML trees [10], [13]–[17]. The models in the other subclass judge the types of webpages according to their layouts [18]–[20].

Filtering models based on autonomous learning are first trained on a certain training set of webpages and then apply certain learning algorithms and statistics, including K-means clustering and term frequency-inverse document frequency (TF-IDF), to cluster the input webpages [21]–[23]. After the clustering process, such a model requires a supplementary algorithm to generate the target classification. It is difficult to determine a suitable threshold for the classification of webpages, and such programs are highly complex [24]–[27].

Certain scholars have fused multiple filtering models to form new filtering models. Some researchers have used the ''fuzzy ontology + support vector machine (SVM) + blacklist'' fusion approach to filter out websites with pornographic content [1], while others have used multistring matching to detect malicious web sites [1], [2] or ''K-nearest neighbor + SVM'' fusion to filter out pornographic websites [3]. The results of these studies show that the fusion of multiple filtering models can improve the precision ratio (*PR*) and recall ratio (*RR*) of the filtering results. Because these studies have used different experimental datasets and the source codes for the fused models are not provided in the literature, these fusion results are difficult to reproduce. However, we have implemented a multistring matching model as a single model for comparative analysis.

Our research team has developed a filtering engine that supports the processing of large sets of web data using multiple individual webpage filtering models and fusions thereof. We call this engine MMFEFWP. MMFEFWP currently supports the following functions:

1) MMFEFWP supports five individual filtering models. These models include a model for filtering pages by strings (MFPS), a model for filtering pages by tags and attributes (MFPTA), a model for filtering pages by trees (MFPT), a model for filtering pages by link ratios (MFPLR), and a model for filtering pages by similarity degree (MFPSD). Among them, MFPS, MFPTA, MFPT, and MFPLR are filtering models based on the features of webpage tags, while MFPSD is a filtering model based on the structures of webpages.

2) MMFEFWP supports the fusion of any two of the five individual filtering models. There are two possible fusion methods. One is to simultaneously satisfy the conditions of both individual models. The other is to satisfy the conditions of one of the two individual models. Such fusion can improve either *PR* or *RR*.

3) MMFEFWP can process millions of webpages in a day. In this study, we use 200,000 webpages collected by crawling the popular online shopping website ''www.jd.com'' as our experimental dataset.

III. ENGINE DESIGN

To develop MMFEFWP, we followed the research route shown in Figure 1. First, we designed the engine functions, the engine architecture, and the engine software. Then, we conducted a performance analysis. We used open-source software such as Spring, jsoup, and HtmlUnit to develop MMFEFWP.

A. ENGINE FUNCTIONS

The functions of MMFEFWP are shown in Figure 2. First, the engine deletes redundant tags from the collected webpages. Then, the engine filters these webpages.

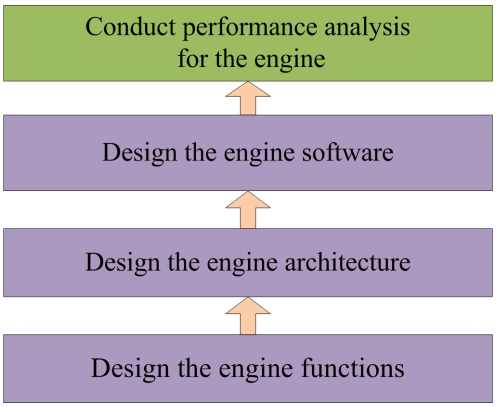


FIGURE 1. Research route of MMFEFWP.

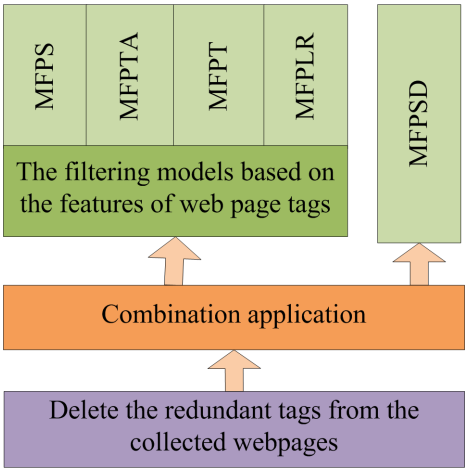


FIGURE 2. Functions of the proposed engine.

When filtering webpages, the engine can use any one of the five individual filtering models or a fusion of any two of them. As our research continues to develop, our engine will support more filtering models and more complex fusions.

MMFEFWP requires a large dataset for experimental analysis. Our research team has developed a multimachine and multithread vertical crawler that can crawl tens of millions of webpages in a day.

B. ENGINE ARCHITECTURE

To realize the fusion function depicted in Figure 2 and to implement the filtering models, we designed MMFEFWP with the hierarchical architecture shown in Figure 3. The technical details of the architecture, including the data processing, filtering algorithms, and open-source software used, are fully described here.

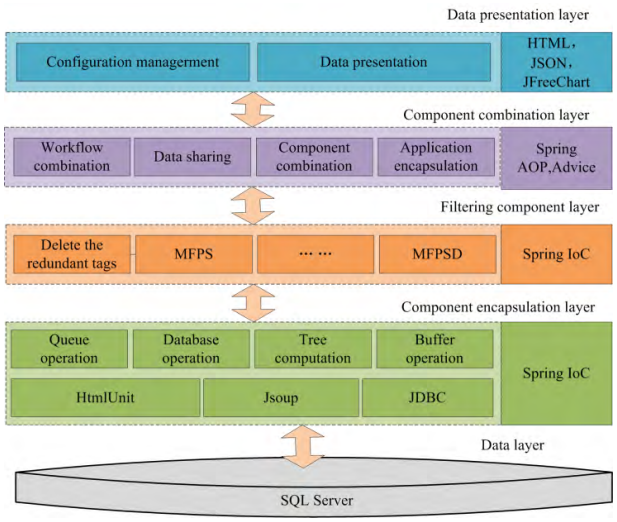


FIGURE 3. Engine architecture.

The engine architecture consists of five layers: a data layer, a component encapsulation layer, a filtering component layer, a component combination layer and a data presentation layer. In the data layer and the component encapsulation layer, webpage queues are implemented as database tables using SQL Server, and jsoup is used as the tool for tree processing. All operations on the queues and all operations on the trees are encapsulated as JavaBeans.

In addition, to speed up data processing, we use a buffer for queue operations. The encapsulated JavaBeans are managed by a Spring container using the Inversion of Control (IoC) design concept. These JavaBeans are encapsulated using two bean types: singleton and prototype.

In the filtering component layer, the following algorithms are implemented in the form of JavaBeans: a tag deleting function, MFPS, MFPTA, MFPT, MFPLR, and MFPSD. In the component combination layer, these components are managed and combined using the aspect-oriented programming (AOP) characteristics of Spring. Spring configuration files are used to configure the pre-Advice and post-Advice parts of the Spring container to manage JavaBeans and form a combined filtering workflow.

In the data presentation layer, HTML and JFreeChart are used to visualize the results of data analysis and show the commodity price analysis curve. By using JavaScript Object Notation (JSON), we can access web services in a simple way to obtain the data for the engine.

C. ENGINE SOFTWARE

According to the architecture design shown in Figure 3, we need a buffer for quickly processing webpages. The design of the filtering function is shown in Figure 4. Based on the original design depicted in Figure 4, we further consider the following two factors.

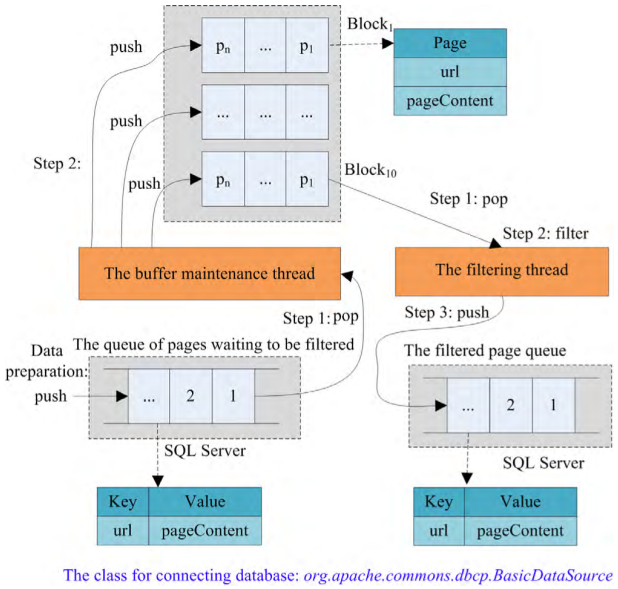


FIGURE 4. Design of the filtering function.

1) QUEUE AND BUFFER STORAGE

The queues are stored as database tables using SQL Server. The queue of pages waiting to be filtered is mapped to one table. The queue of filtered pages is mapped to another table. We use the BasicDataSource class in Spring to encapsulate the data sources. The elements in these queues have the structure Key–Value, where Key is the URI of the webpage and Value is the content of the webpage. The elements in the buffer queue are all Page objects, each of which has two properties: the ''url'', which is the URI of the webpage, and the ''pageContent'', which is the content of the webpage. We assume that the buffer contains k queues, a queue has n elements, the length of a URI is l bytes, and the length of the content of a page is m bytes. Then, the buffer size BS is calculated using the following formula:

$$BS = k \times n \times (l + m) \tag{1}$$

We set $k = 10$, $n = 10$, $l = 300$, and $m = 40,000$; then, the buffer size is:

$$\begin{aligned} BS &= k \times n \times (l + m) \\ &= 10 \times 10 \times (300 + 40,000) \\ &= 4,030,000 \text{ bytes} \end{aligned}$$

Therefore, the buffer occupies approximately 4 MB of memory space, which is acceptable for the current configurations of mainstream servers.

2) FILTERING PROCESS

During the data preparation stage, the crawler pushes the crawled webpages into the queue of pages waiting to be filtered. The buffer maintenance thread is used to provide the datasets of the webpages to the filtering thread. The procedure executed by the buffer maintenance thread is provided in algorithm 1.

Algorithm 1 Buffer Maintenance Algorithm

```
    /* While the queue of pages waiting to be filtered is
not empty */
1: While (QueueWaiting is not empty)
    //Traverse the buffer to find an empty block
2:      For i = 1 to 10 step 1
        //If a block in the buffer is empty
3:      If Block_i is empty Then
            /* Pop out n webpages from
               QueueWaiting into block */
4:          popup(QueueWaiting, block, n)
            // Push block into Block_i
5:          push(Block_i, block)
6:      End If
7:    End For
8: End While
```

According to algorithm 1, once the buffer maintenance thread discovers an empty block in the buffer, it pops out n webpages from the queue of pages waiting to be filtered and then pushes these webpages into the empty block found.

The procedure executed by the filtering thread is given in algorithm 2.

According to algorithm 2, once the filtering thread finds a data block that is not empty, it uses either a single filtering model or a fused filtering model to filter the webpages in that data block. Then, the thread pushes the filtering results into the filtered page queue.

D. FUSION RULES

At present, MMFEFWP supports the fusion of two of the five individual filtering models. We derived the rules describing how the *PR* and *RR* change for two methods of fusion, which are given below:

1) Fusion rule 1. The conditions of both individual models are simultaneously satisfied. If we denote the individual

Algorithm 2 Webpage Filtering Algorithm

```
    /* While the queue of pages waiting to be filtered is not
    empty and the buffer is not empty
*/
1: While (QueueWaiting is not empty and buffer is
    not empty)
    /* Traverse the buffer to find a block that is not
     empty */
2:    For i = 1 to 10 step 1
            // If a block in the buffer is not empty
3:    If Block_i is not empty Then
            // Pop out Block_i in buffer into block
4:      popup(buffer, Block_i, block)
            /* Filter the webpages using the filtering
               models */
5:      filterResult=filterWebPages(block)
            /* Push the filtering result into the
               filtered page queue */
6:      push(filterResult, QueueFiltered)
7:    End If
8:  End For
9: End While
```

filtering conditions of the two fused filtering models by *filterModel1.conditions* and *filterModel2.conditions*, respectively, then the conditions to be satisfied for this fusion method are expressed as follows:

filterModel1.conditions **and** *filterModel2.conditions*

Theoretically, we believe that the filtering performance after fusion should be described by the following two rules.

First, the *RR* of the fused results should be less than or equal to the smaller of the *RR*s of the two individual filtering models. This rule is expressed as follows:

$$RR \leq \text{Min}(RR_1, RR_2) \tag{2}$$

Second, the *PR* of the fused results should be greater than or equal to the larger of the *PR*s of the two individual filtering models. This rule is expressed as follows:

$$PR \geq \text{Max}(PR_1, PR_2) \tag{3}$$

Here, the formulas for calculating *RR* and *PR* are as follows:

$$RR = \frac{TAF}{TUF} \times 100\% \tag{4}$$

where *TAF* is the number of target pages obtained after filtering and *TUF* is the number of target pages in the pages to be filtered;

$$PR = \frac{TAF}{AF} \times 100\% \tag{5}$$

where *AF* is the number of pages obtained after being filtered.

We derive the two rules above as follows. For *RR*, the number of target pages in the pages to be filtered is fixed, and the

number of target pages obtained after filtering should either decrease or remain unchanged. Therefore, the calculated *RR* should also either decrease or remain unchanged.

For *PR*, the number of target pages obtained after filtering and the number of pages obtained after filtering should either be reduced or remain unchanged. The range of variation of the number of pages obtained after filtering is greater than the range of variation of the number of target pages obtained after filtering. Therefore, the calculated *PR* should either increase or remain unchanged. Moreover, the change in *PR* should be smaller than the change in *RR*. These rules will be shown to hold in the subsequent experimental analysis.

2) Fusion rule 2. The conditions of only one of the two individual models must be satisfied. The conditions to be satisfied for this fusion method are expressed as follows:

filterModel1.conditions **or** *filterModel2.conditions*

Theoretically, we believe that the filtering performance after fusion should be described by the two following rules.

First, the *RR* of the fused results should be greater than or equal to the larger of the *RR*s of the two individual filtering models. This rule is expressed as follows:

$$RR \geq \mathrm{Max}(RR_1, RR_2) \tag{6}$$

Second, the *PR* of the fused results should be less than or equal to the smaller of the *PR*s of the two individual filtering models. This rule is expressed as follows:

$$PR \leq \mathrm{Min}(PR_1, PR_2) \tag{7}$$

The derivation of these rules is similar to that for fusion method 1, and so it is not repeated here. To determine which fusion method should be used in practice, we proposed the following selection principles. If the *RR*s of the two individual filtering models are both acceptable and the *PR*s are unacceptable, *PR* needs to be improved. Thus, we suggest choosing fusion rule 1. If the *PR*s of the two individual filtering models are both unacceptable and the *RR*s are also unacceptable, *RR* needs to be improved. Thus, we suggest choosing fusion rule 2. To enable this determination, an acceptance threshold can be set for *RR* and *PR* (e.g., 85%).

IV. PERFORMANCE ANALYSIS

To verify the functions of MMFEFWP and the rules described above, we used 200,000 webpages crawled from the popular online shopping website "www.jd.com" as our experimental dataset. The dataset contains 57,570 pages with detailed information. The filtering goal is to find these detailed information pages.

A. EXPERIMENTAL ENVIRONMENT

We used a laptop as the experimental environment, with "SQL Server 2017 Desktop Edition" as the database software and "Eclipse Java EE IDE for Web Developers Neon.3" as the development software. The specifications of the computer hardware and the operating system are listed in Table 1.

TABLE 1. Computer configuration used in the experiments.

Environment parameter	Configuration
Computer model	ThinkPad T460s
CPU	Intel Core i5-6200U CPU @ 2.30 GHZ
Memory	8 GB of RAM
Hard disk	512 GB solid-state drive
Operating system	Windows 10

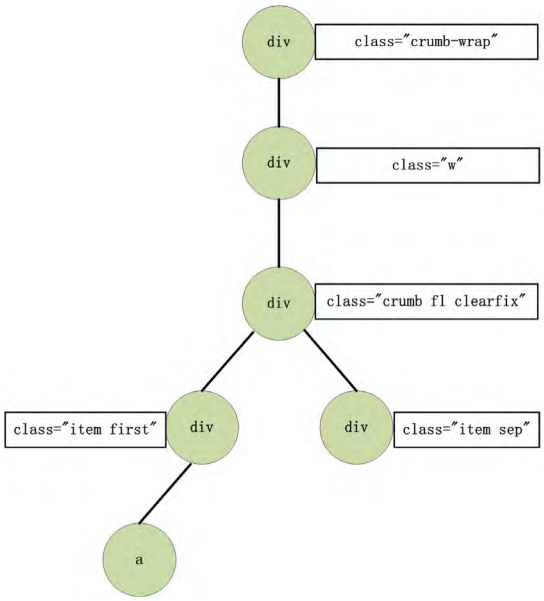


FIGURE 5. Structure tree used for filtering.

B. EXPERIMENTAL ANALYSIS OF THE INDIVIDUAL FILTERING MODELS

The effects of the individual filtering models were experimentally analyzed. The main configuration of MMFEFWP for each model is listed in Table 2. In each experiment, the same 200,000 pages were filtered. The experimental results are shown in Figure 6 and Figure 7. The performances of the filtering models were assessed in terms of the processing time per thousand pages and the cumulative processing time. The accuracies of the filtering models were assessed in terms of *RR* and *PR*. For effective calculation in real experiments, the number of detailed information pages obtained after being filtered is considered as *TAF* and the number of target pages in the pages to be filtered is considered as *TUF*.

In Figure 6 and Figure 7, the abscissa represents the number of webpages filtered, whereas the ordinate is the *RR* or *PR* value. Because the trends in the *RR* and *PR* values can be clearly seen in Figure 6, detailed explanations are not given below.

C. FUSED APPLICATION OF THE FIVE FILTERING MODELS

The purpose of combining individual filtering models is to improve the *RR* and *PR* of the filtering results. As seen

TABLE 2. Main configurations of MMFEFWP.

Filtering model	Main configuration
MFPS	The following conditions must be satisfied at the same time. **Condition 1.** Each webpage obtained after filtering must include one of the following strings: "price", "flash purchase price", "Jingdong price", "exclusive price", or "price spike". This condition is expressed as follows: *page*.containsString("price") **or** *page*. containsString ("flash purchase price") **or** *page*.containsString("Jingdong price") **or** *page*.containsString("exclusive price") **or** *page*.containsString("price spike") **Condition 2.** Each webpage obtained after filtering must include the string "distribution". This condition is expressed as follows: *page*.containsString("distribution") **Condition 3.** Each webpage obtained after filtering must include one of the following strings: "commodity details" or "commodity introduction". This condition is expressed as follows: *page*.containsString("commodity details") **or** *page*.containsString("commodity introduction")
MFPTA	The following condition must be satisfied. **Condition 1.** Each webpage obtained after filtering must include the tag , and the value of the "class" attribute of the tag must be "crumb-wrap". This condition is expressed as follows: *page*.containsTag(*divTag*) **and** *page.divTags*.containsAttribute(*classAttribute*)
MFPT	The following condition must be satisfied. **Condition 1.** Each webpage obtained after filtering must have a structure tree such as that shown in Figure 5. This condition is expressed as follows: *page*.containsTree(*treeFigure5*)
MFPLR	The following condition must be satisfied. **Condition 1.** The link ratio threshold is set to 0.25. This condition is expressed as follows: *page*.linkRatioOut(0.25)
MFPSD	The following conditions must be satisfied at the same time. **Condition 1.** The tags in each webpage tree must include only the tags and . This condition is expressed as follows: *page*.onlyContainTags(*divTagAndspanTag*) **Condition 2.** The similarity between each webpage and a template webpage is calculated as the ratio of the number of identical nodes between the trees to the total number of nodes in the smaller of the two trees. The condition for two nodes to be identical is that the node name and the class attribute must have the same values. The similarity threshold is set to 0.5. This condition is expressed as follows: *page*.similarityOut(0.5)

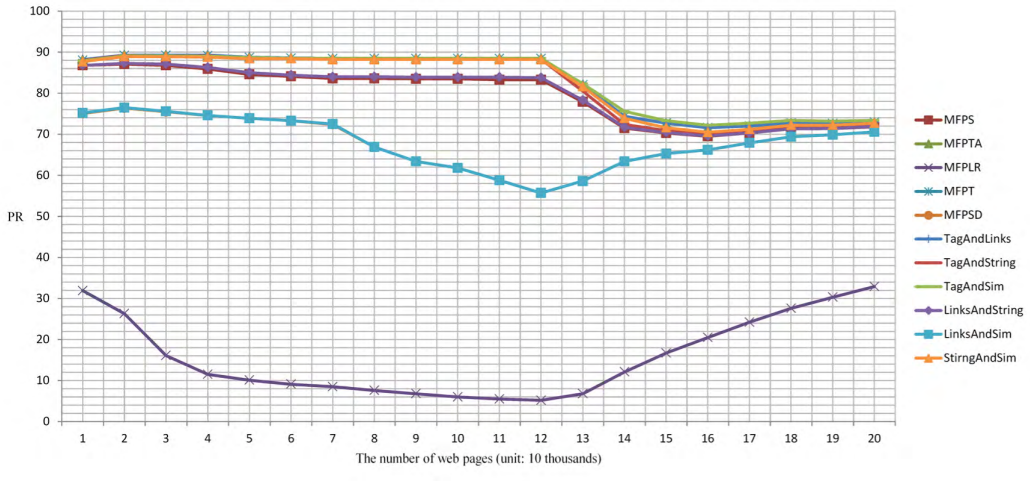


FIGURE 6. *PR*s of the individual filtering models and fusion models.

from the above analysis, the *RR*s of the five filtering models are acceptable, but their *PR*s are not ideal. Therefore, we adopt the following approaches in our fusion analysis.

1) Choose a fusion method. To improve *RR*, fusion rule 1 should be chosen.

2) Merge two filtering models. Considering that the effects of the MFPTA and MFPT models are very similar,

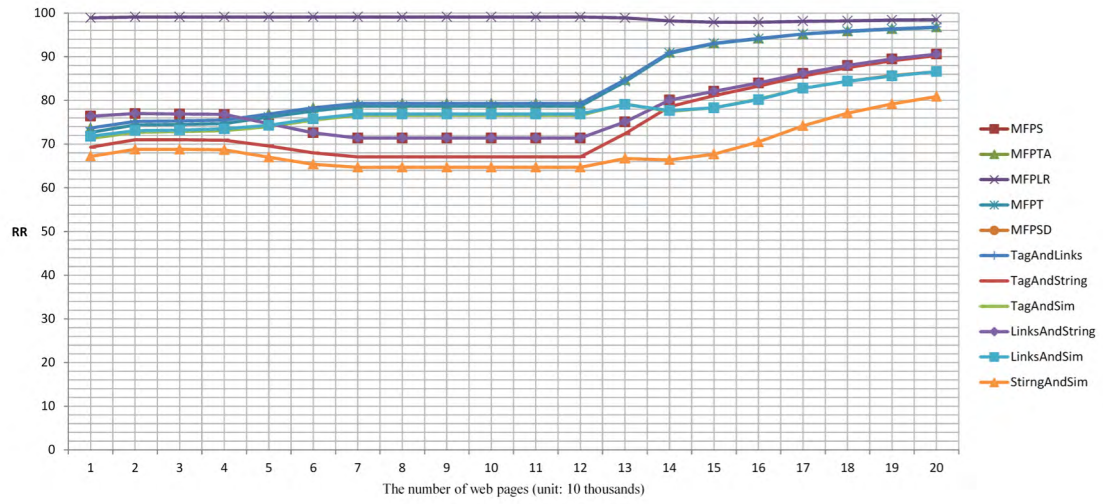


FIGURE 7. *RR*s of the individual filtering models and fusion models.

TABLE 3. *RR*s and *PR*s of the models when filtering nearly 200,000 webpages.

Filtering model	*TUF*	*AF*	*TAF*	*RR*	*PR*
MFPS	57570	72643	52175	90.6%	71.8%
MFPTA	57570	76441	55725	96.8%	72.9%
MFPLR	57570	172694	56734	98.5%	32.9%
MFPT	57570	76407	55692	96.7%	72.9%
MFPSD	57570	70643	49843	86.6%	70.6%
TagAndLinks	57570	76423	55728	96.8%	72.9%
TagAndString	57570	71098	51987	90.3%	73.1%
TagAndSim	57570	67899	49858	86.6%	73.4%
LinksAndString	57570	72547	52176	90.6%	71.9%
LinksAndSim	57570	70632	49873	86.6%	70.6%
StringAndSim	57570	64062	46586	80.9%	72.7%

with MFPTA having a higher efficiency and a slightly higher *RR*, the MFPTA filtering model is considered representative of both these models for the remainder of the experimental analysis.

Accordingly, we consider the following six fused filtering models.

1) The fusion of the MFPTA and MFPLR filtering models, denoted by TagAndLinks. The filtering conditions of TagAndLinks are that the conditions of MFPTA and MFPLR are all satisfied.

2) The fusion of the MFPTA and MFPS filtering models, denoted by TagAndString. The filtering conditions of TagAndString are that the conditions of MFPTA and MFPS are all satisfied.

3) The fusion of the MFPTA and MFPSD filtering models, denoted by TagAndSim. The filtering conditions of TagAndSim are that the conditions of MFPTA and MFPSD are all satisfied.

4) The fusion of the MFPLR and MFPS filtering models, denoted by LinksAndString. The filtering conditions of LinksAndString are that the conditions of MFPLR and MFPS are all satisfied.

5) The fusion of the MFPLR and MFPSD filtering models, denoted by LinksAndSim. The filtering conditions of LinksAndSim are that the conditions of MFPLR and MFPSD are all satisfied.

6) The fusion of the MFPS and MFPSD filtering models, denoted by StringAndSim. The filtering conditions of StringAndSim are that the conditions of MFPS and MFPSD are all satisfied.

The *RR*s and *PR*s of the six fused models above are shown in Figure 6 and Figure 7.

D. DISCUSSION

To facilitate a comparative analysis, Table 3 lists the *RR*s and *PR*s achieved by the models when the number of filtered webpages is nearly 200,000. The curve for the fused TagAndLinks model in Figure 6 and Figure 7 is similar to that for the individual MFPTA model. When the number of webpages filtered is nearly 200,000, the *RR* and *PR* of the fused TagAndLinks model are the same as those of the MFPTA model, whereas the *RR* and *PR* of the fused TagAndString model are 90.3% and 73.1%, respectively. We abstain

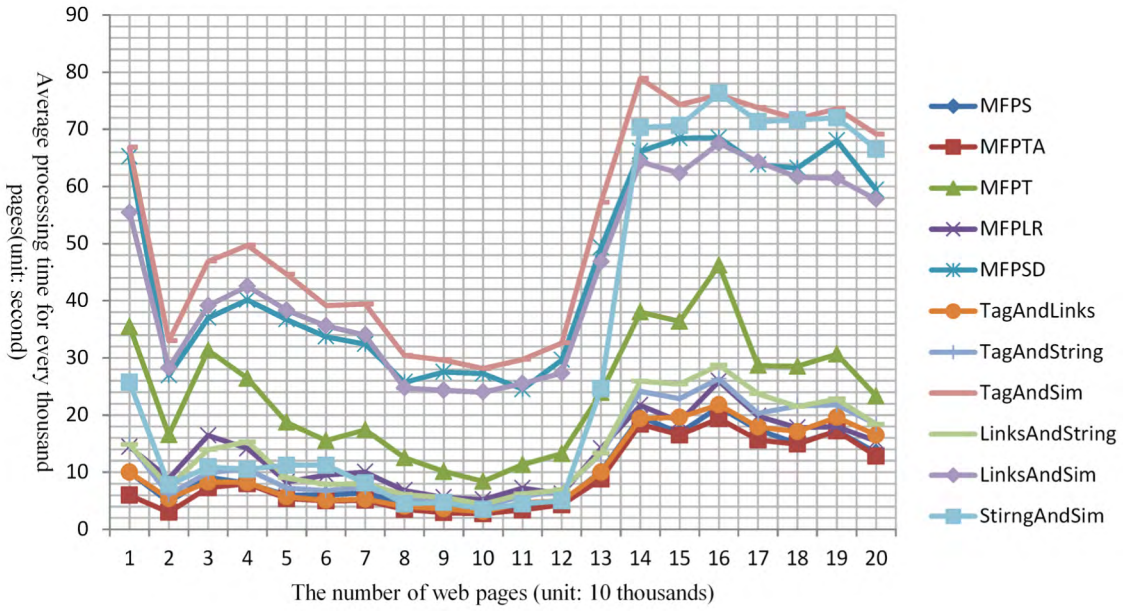


FIGURE 8. Average processing time for every thousand pages of the filtering models.

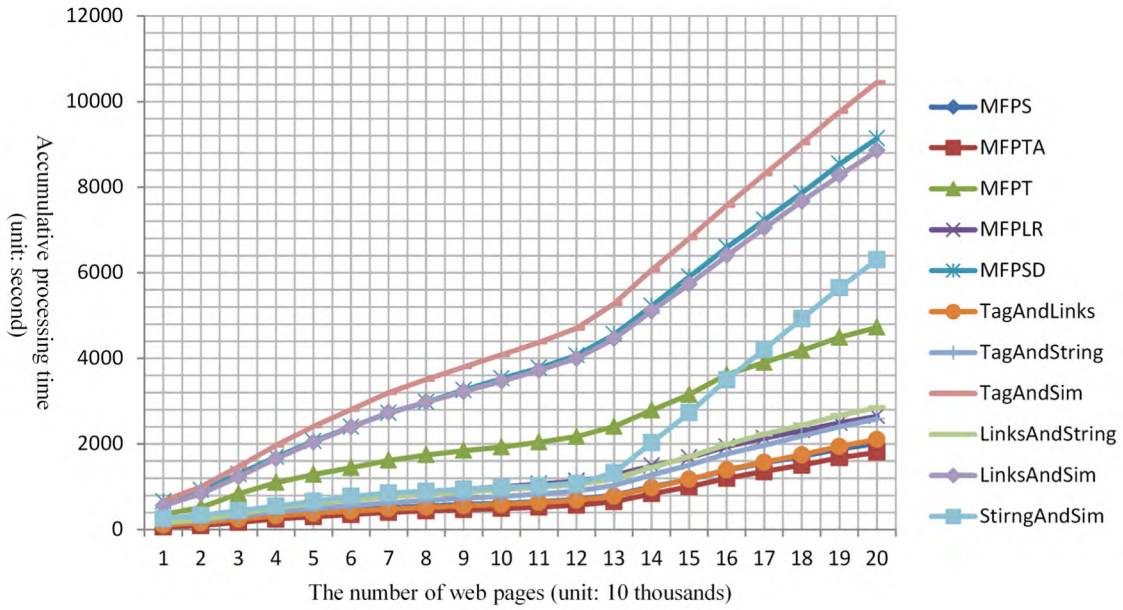


FIGURE 9. Accumulative processing time of the filtering models.

from reiterating the *RR*s and *PR*s of the other four fused filtering models here.

The curve for MFPTA is similar to that for MFPT. The similarity in the *RR*s and *PR*s of these two filtering models can be explained by considering the conditions applied in these models. As seen from Table 2, the conditions "*page*.containsTag(*divTag*) **and** *page.divTags*.containsAttribute(*classAttribute*)" are included in Condition 1 of MFPTA and Condition 1 of MFPT.

The execution efficiencies of the filtering models are shown in Figure 8 and Figure 9. In Figure 8, the ordinate is the average processing time in seconds for every thousand pages. In Figure 9, the ordinate is the cumulative processing time for all pages in seconds.

In general, the MFPTA model has the highest speed. This model takes only 1802.4 seconds, or approximately 30 minutes, to process 200,000 pages. The TagAndSim model is the slowest, taking 10447.4 seconds, or approximately 174.1 minutes, to process 200,000 pages; this is approximately six times slower than the MFPTA model.

From Figures 6–9 and Table 3, we can infer the following rules.

1) Among the six fused models, if the fused model was obtained by fusing MFPLR with another filtering model, then the graph for the fused model is biased towards that for the other model. For example, the graph for the fused TagAndLinks model is biased towards that for the individual MFPTA model. According to our analysis, the reason for this behavior is that the set of results produced by MFPLR is too large. When the number of filtered webpages is nearly 200,000, the number of MFPLR results is approximately 173,000, indicating that the MFPLR model has no obvious filtering effect.

2) The correctness of formula 2 and formula 3 is verified. *RR* decreases significantly with the fusions considered here, while *PR* does not markedly increase. For example, when the number of filtered webpages is nearly 200,000, the *RR* and *PR* of the fused StringAndSim model are 80.9% and 72.7%, respectively. The *RR* of this fused model is decreased by 9.7 percentage points and 5.7 percentage points compared with those of the individual MFPS and MFPSD models, respectively, while *PR* is correspondingly increased by only 0.9 percentage points and 2.1 percentage points, respectively.

We abstain here from describing fused models obtained using fusion rule 2 and from demonstrating the correctness of formula 6 and formula 7. As discussed above, the developed MMFEFWP engine supports five individual filtering models and their fusion, but the following three problems still remain.

1) We could not find a filtering model with both high *RR* and high *PR*. MMFEFWP supports several individual filtering models with high *RR*s, the lowest value of *RR* being 86.6%. However, the *PR*s of these individual models are not ideal, with the highest being only 72.9%. Even when two individual filtering models are fused, the highest *PR* is increased to only 73.4%. One possible way to solve this problem is to use an artificial intelligence algorithm, such as a deep neural network.

2) The number of individual filtering models supported by the engine is low. At present, MMFEFWP supports only five individual filtering models, and the *PR*s of these models are not high. More individual filtering models need to be supported.

3) The fusion capability supported by the engine is limited. Currently, MMFEFWP supports the fusion of only two individual filtering models. In the future, the fusion of three or more filtering models should be supported by the engine. It will also be necessary to study the rules governing how the *PR*s and *RR*s change with such complex fusions.

Our research team will address these three problems in our future work.

V. CONCLUSION

We have proposed the MMFEFWP architecture for the filtering of large datasets of webpages crawled from websites. Our proposed multimodel fusion engine for webpage filtering can extract target webpages using multiple models, including models based on strings, trees, link ratios, similarity degrees, and tags and attributes. We downloaded 200,000 pages from the popular online shopping website "www.jd.com" for use in experimental analysis. Our results show that fused filtering models achieve better *PR* values than the individual models; for example, the *PR* of the TagAndLinks model is 72.9%, which is considerably higher than the *PR* of 32.9% achieved by the individual MFPLR model. Although the *PR*s of the fused models are still not high, it is obvious that two-model fusion results in improved performance compared with the individual filtering models. Through fusion, the *PR* of an individual filtering model can be improved if its *RR* is acceptable, or the *RR* can be improved if the *PR* is acceptable.

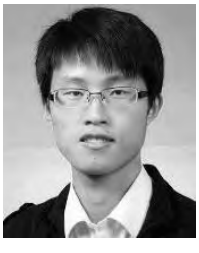
In summary, our proposed MMFEFWP engine can filter a large number of webpages. This engine can be profitably used in engineering practice. In the future, in addition to solving the three problems mentioned in the previous section, we will integrate MMFEFWP with a crawler and a web system for displaying data.

REFERENCES

[1] F. Ali *et al.*, "A fuzzy ontology and SVM–based Web content classification system," *IEEE Access*, vol. 5, pp. 25781–25797, 2017.
[2] A. Díaz-Manríquez, A. B. Ríos-Alvarado, J. H. Barrón-Zambrano, T. Y. Guerrero-Melendez, and J. C. Elizondo-Leal, "An automatic document classifier system based on genetic algorithm and taxonomy," *IEEE Access*, vol. 6, pp. 21552–21559, 2018.
[3] V. K. Bhalla and N. Kumar, "An efficient scheme for automatic Web pages categorization using the support vector machine," *New Rev. Hypermedia Multimedia*, vol. 22, no. 3, pp. 223–242, 2016.
[4] A. Ahmadi, M. Fotouhi, and M. Khaleghi, "Intelligent classification of Web pages using contextual and visual features," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1638–1647, 2011.
[5] A. I. Saleh, M. F. Al Rahmawy, and A. E. Abulwafa, "A semantic based Web page classification strategy using multi-layered domain ontology," *World Wide Web*, vol. 20, no. 5, pp. 939–993, 2017.
[6] J.-H. Lee, W.-C. Yeh, and M.-C. Chuang, "Web page classification based on a simplified swarm optimization," *Appl. Math. Comput.*, vol. 270, pp. 13–24, Nov. 2015.
[7] Y. Du, Q. Pen, and Z. Gao, "A topic-specific crawling strategy based on semantics similarity," *Data Knowl. Eng.*, vol. 88, pp. 75–93, Nov. 2013.
[8] M. Sahami and T. D. Heilman, "A Web-based kernel function for measuring the similarity of short text snippets," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 377–386.
[9] S. Kohli, S. Kaur, and G. Singh, "A Website content analysis approach based on keyword similarity analysis," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol.*, vol. 1, Dec. 2012, pp. 254–257.
[10] D. A. Popescu and D. Radulescu, "Approximately similarity measurement of Web sites," in *Proc. Int. Conf. Neural Inf. Process.*, 2015, pp. 624–630.
[11] H. Li, Z. Xu, T. Li, G. Sun, and K.-K. R. Choo, "An optimized approach for massive Web page classification using entity similarity based on semantic network," *Future Gener. Comput. Syst.*, vol. 76, pp. 510–518, Nov. 2017.
[12] E. Ilbahar and S. Cebi, "Classification of design parameters for E-Commerce Websites: A novel fuzzy Kano approach," *Telematics Inform.*, vol. 34, no. 8, pp. 1814–1825, 2017.
[13] G. S. Reddy and D. R. V. Krishnaiah, "Clustering algorithm with a novel similarity measure," *J. Comput. Eng.*, vol. 4, no. 6, pp. 37–42, 2012.

[14] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, Dec. 1997.
[15] Z. Deng, J. Zhang, and T. He, "Automatic combination technology of fuzzy CPN for OWL-S Web services in supercomputing cloud platform," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 31, no. 7, p. 1759010, 2017.
[16] Y. Du and Y. Hai, "Semantic ranking of Web pages based on formal concept analysis," *J. Syst. Softw.*, vol. 86, no. 1, pp. 187–197, 2013.
[17] X. Xie and B. Wang, "Web page recommendation via twofold clustering: Considering user behavior and topic relation," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 235–243, 2018.
[18] A. S. Bozkir and E. A. Sezer, "Layout-based computation of Web page similarity ranks," *Int. J. Hum. Comput. Stud.*, vol. 110, pp. 95–114, Feb. 2018.
[19] C. Fang and B. Y. Liu, "Exploration of Web page structural patterns based on request dependency graph decomposition," *Int. J. Digit. Crime Forensics*, vol. 8, no. 4, pp. 1–13, 2016.
[20] J. Kumar, P. Ye, and D. Doermann, "Structural similarity for document image classification and retrieval," *Pattern Recognit. Lett.*, vol. 43, pp. 119–126, Jul. 2014.
[21] G. Kou and C. Lou, "Multiple factor hierarchical clustering algorithm for large scale Web page and search engine clickstream data," *Ann. Oper. Res.*, vol. 197, no. 1, pp. 123–134, 2012.
[22] T. T. S. Nguyen, H. Y. Lu, and J. Lu, "Web-page recommendation based on Web usage and domain knowledge," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2574–2587, Oct. 2014.
[23] A. A. AbdulHussien, "Comparison of machine learning algorithms to classify Web pages," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 11, pp. 205–209, 2017.
[24] R. Malhotra and A. Sharma, "Quantitative evaluation of Web metrics for automatic genre classification of Web pages," *Int. J. Syst. Assurance Eng. Manage.*, vol. 8, no. 2, pp. 1567–1579, 2017.
[25] K. Chinniyan, S. Gangadharan, and K. Sabanaikam, " Semantic similarity based Web document classification using support vector machine," *Int. Arab J. Inf. Technol.*, vol. 14, no. 3, pp. 285–292, 2017.
[26] A. R. W. Sait and T. Meyyappan, "An automated Web page classifier and an algorithm for the extraction of navigational pattern from the Web data," *J. Web Eng.*, vol. 16, nos. 1–2, pp. 126–144, 2017.
[27] F. Ali *et al.*, "A fuzzy ontology and SVM-based Web content classification system," *IEEE Access*, vol. 5, pp. 25781–25797, 2017.

ZIYUN DENG was born in Shuangfeng County, China, in 1979. He received the B.Eng. degree in computer science and technology from Central South University, Changsha, China, in 2003, and the M.S. degree in software engineering and the Ph.D. degree in control science and engineering from Hunan University, Changsha, in 2006 and 2016, respectively. From 2013 to 2016, he was a Professor with the Hunan Vocational College of Modern Logistics. Since 2016, he has been a Professor with the Changsha Commerce and Tourism College. His recent research interests lie in high-performance computing and logistics information technology.



TINGQIN HE received the M.S. degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He is currently pursuing the Ph.D. degree with the College of Information Science and Engineering, Hunan University. His research interests include data mining, cloud computing, and big data analysis.

WEIPING DING (M'16) received the Ph.D. degree in applied computing from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2013. He was a Visiting Researcher with the University of Lethbridge, Lethbridge, AB, Canada, in 2011. From 2014 to 2015, he was a Post-Doctoral Researcher with the Brain Research Center, National Chiao Tung University, Hsinchu, Taiwan. In 2016, he was a Visiting Scholar with the National University of Singapore, Singapore. From 2017 to 2018, he was a Visiting Scholar with the University of Technology Sydney, Ultimo, NSW, Australia. He has authored over 50 papers in flagship journals and conference proceedings as the first author; to date, he holds 10 approved invention patents among a total of 18 issued patents. His current research interests include data mining, machine learning, and granular computing.

Dr. Ding was a recipient of the NUAA Excellent Doctoral Dissertation Award in 2015, the Best Paper of ICDMA 2015, Hong Kong, and two Chinese Government Scholarships for Overseas Studies in 2011 and 2016. He received the Computer Education Excellent Paper Award (First Prize) from the National Computer Education Committee of China in 2009. He was an Excellent Young Teacher (Qing Lan Project) of Jiangsu Province in 2014 and a High-Level Talent (Six Talent Peak) of Jiangsu Province in 2016. He served/serves as an Associate Editor for the IEEE Transactions on Fuzzy Systems, *Information Sciences*, and *Swarm and Evolutionary Computation*.

ZEHONG CAO (M'18) received the B.Eng. degree in electronic and information engineering from Northeastern University, Shenyang, China, in 2012, the M.S. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2013. He received the dual Ph.D. degrees in Information Technology from UTS and Electrical and Control Engineering from National Chiao Tung University in 2017. He is currently a Post-Doctoral Research Fellow with the Center for Artificial Intelligence, UTS. His current research interests include data science, human–machine interfaces, computational intelligence, pattern recognition, machine learning, and clinical applications.

Dr. Cao was a recipient of the CAMP Award in 2018, the UTS CAI Best Paper Award in 2017, the UTS FEIT Publication Award in 2017, the UTS President's Scholarship in 2015, and the NCTU & Songshanhu Scholarship in 2013. He serves as an Associate Editor for the IEEE Access and as a member for the editorial boards of several journals, including *Advances in Robotics and Automation* and the *International Journal of Sensor Networks and Data Communications*. He was invited to give oral presentations at the IEEE-FUZZY in 2017, the IJCNN in 2015, and the BME Annual Conference of Taiwan in 2015.